Fidelity and Efficiency of Knowledge Representations for Intelligent Tutoring Systems

Stellan Ohlsson[1] and Antonija Mitrovic[2]

[1]University of Illinois at Chicago, Chicago, USA

[2] University of Canterbury, Christchurch, New Zealand

Author contact information:

Professor Stellan Ohlsson
University of Illinois at Chicago
Department of Psychology
1007 West Harrison Street
MC 285, Room BSB 1050C
Chicago, IL 60607
USA

Email: stellan@uic.edu
Fax: (312) 413-4122
Voice mail: (312) 996-6643

Keywords: cognitive fidelity, computational efficiency, constraint-based tutoring, error detection, feedback, intelligent tutoring systems, knowledge representation, model tracing

Author Note

The preparation of this article was supported, in part, by Grant No. N000140710040 from the Office of Naval Research (ONR), US Navy, to Barbara DiEugenio and Stellan Ohlsson, and, in part, by the University of Canterbury grant U6532 and eCDF grant 592 to the second author. No endorsement should be inferred. We wish to thank our colleagues in the Intelligent Computer Tutoring Group (ICTG) at Canterbury University as well as the Deep Learning Tutoring Group at the University of Illinois at Chicago for stimulating discussions regarding tutoring.

Communication regarding this publication can be sent to Stellan Ohlsson, University of Illinois at Chicago, Department of Psychology, 1007 West Harrison Street, Chicago, IL 60607, USA, or to Antonija Mitrovic, Department of Computer Science and Software Engineering, University of Canterbury, Private Bag 4800, Christchurch, New Zealand. Electronic mail may be sent to stellan@uic.edu or tanja.mitrovic@canterbury.nz.

Abstract

Intelligent tutoring systems differ from ordinary educational software in that they contain explicit representations of both the target subject matter and the student's knowledge, and that pedagogical decisions about which feedback to give, which practice problem to propose, etc. are based, in part, upon those knowledge representations. We argue that a suitable representation should satisfy two criteria that we call *fidelity* and *efficiency*. These criteria are developed with respect to *constraints*, a particular representation designed to satisfy both. A comparison between constraints and production rules with respect to these two criteria highlights the advantages of the constraint-based approach over the rule-based, model-tracing approach.

Fidelity and Efficiency of Knowledge Representations for Intelligent Tutoring Systems

Stellan Ohlsson and Antonija Mitrovic

Tutoring technology, in combination with world-wide network technology, has the potential to provide quality instruction for learners everywhere. Progress has been uneven. While network technology has surged ahead to become an everyday presence, tutoring technology has moved only a short distance beyond its beginnings 30 years ago (Sleeman & Brown, 1982). The key difference between intelligent tutoring systems (ITSs) and ordinary educational software is that ITSs operate on the basis of explicit representations of the target subject matter knowledge and of the student's knowledge. The central principle that underpins intelligent tutoring research is that such representations provide a basis for better pedagogical decisions and potentially for instruction that can match the effectiveness of human one-on-one tutoring.

The choice of knowledge representation is therefore a central issue for intelligent tutoring (in addition to interface design, curriculum development and pedagogy, issues shared with other instructional technologies). On the basis of what should such a choice be made? As elsewhere in the field of artificial intelligence (AI), different knowledge representations exhibit different profiles of advantages and disadvantages. The field of ITS research might benefit from a systematic way of assessing knowledge representations.

We discuss two dimensions along which knowledge representations might differ and which we claim are of central importance in ITS research. The dimensions reflect the double nature of an ITS as a cognitive and a computational artifact. The first dimension, or clusters of dimensions, is *cognitive fidelity*: To what extent, and in which aspects, does a particular knowledge representation mirror the cognition of learners and teachers? That is, to what extent does the ITS model human cognition in general and learning in particular? What are the advantages of such fidelity in an ITS? What are the consequences of lack of such fidelity?

The second dimension, or cluster of dimensions, is *efficiency*. An ITS is a complicated artifact and if we are to provide the world with ITSs for every conceivable instructional topic, we need to adopt a tutoring technology that facilitates rapid implementation, evaluation and deployment (*implementation efficiency*). In addition, an implemented ITS lives in the real world, with real students banging on real key boards in real time. ITS technology had better be fast and robust before being let loose on the networks (*computational efficiency*).

The choice of knowledge representation can affect any of these dimensions. In the following, we first introduce the constraint-based approach to knowledge representation. In the main section, we explore the two dimensions of fidelity and efficiency in the context of the constraint representation. We compare this representation to production systems as they are used in model-tracing systems, a prominent alternative, using our dimensions to guide the comparison. At the end, we offer some general conclusions.

CONSTRAINT-BASED MODELING

Cognitive scientists draw upon several standard knowledge representation formalisms to explain and model cognitive phenomena: Feature vectors, propositions, semantic networks, schemas, production rules, Horn clauses, STRIPS operators, goal trees, goal stacks and so on (Markman, 1999). Each of these representations captures some feature or features of knowledge well while capturing others only partially or not at all. For example, semantic networks capture the interrelatedness of all conceptual knowledge but fail to capture local structure; schemas exhibit exactly the opposite profile. Production rules capture the situated character of practical knowledge, while goal-trees capture the hierarchical organization of action. Rule-based representations consist of a declarative (propositional) knowledge base, a rule set and an interpreter that executes the rules vis-à-vis the knowledge base. This approach provides flexibility in execution and is thus particularly suitable for domains in which the ordering of problem solving steps can vary. No representation captures all aspects of knowledge and the set is eclectic. There is no principled reason to focus on precisely these knowledge representations. They are merely the ones cognitive scientists have thought of so far and know how to implement in computer programs. Inventing novel knowledge representations is one way for the cognitive sciences to advance.

In prior work, we have developed a representation for declarative knowledge with rather different properties than the standard, propositional representations. We refer to the unit of knowledge in this representation as a *constraint* (Ohlsson & Rees, 1991a, 1991b), although this term means something slightly different in our usage than it does in the context of standard constraint-satisfaction models (Freuder & Mackworth, 1994; Lehler 1988; Miguel, 2004). Constraints were invented to support different cognitive operations than those supported by propositions. The latter were originally developed by logicians to support inference, explanation and prediction. Constraints, in contrast, support evaluation and judgment. The latter processes are central for both performance and learning, because the learner needs to evaluate his or her progress, judge the outcomes of his or her actions and decide whether such-and-such a state of affairs is satisfactory and so on.

A constraint C is a binary pair, C = <R, S>, in which R is *the relevance criterion* and S is *the satisfaction criterion*. R specifies the circumstances (set of situations) in which the constraint is relevant, while S specifies the circumstances (set of situations) in which the constraint is satisfied.

The semantics of a constraint is that *if R is satisfied in the current situation, then S ought to be satisfied also (or else something has gone wrong)*. A constraint is thus an atom of evaluative knowledge. It enables a judgment to be made as to whether the state of affairs that the agent has reached is acceptable or whether it should be regarded as unacceptable. A common mistake is to regard S as an inference to be made when R is the case; this is not the intent. A relevance criterion is not grounds for believing or concluding that the associated satisfaction criterion is true. The constraint encodes the norm that if R is the case, the S ought to be the case as well, not a conditional assertion that S is in fact the case.

If R is the case, the conclusion that follows upon the execution of the constraint depends on the status of S. If S is satisfied, then the conclusion is that *the situation at*

hand satisfies C. If S is not satisfied, the conclusion is that *the situation at hand does not satisfy C*. We refer to this as a *constraint violation*. If the relevance criterion is not satisfied, then the constraint is irrelevant and no conclusion follows, regardless of the status of S.

In everyday discourse, evaluative knowledge is often expressed in what seems to be declarative sentences that correspond to propositions. Traffic regulations provide examples. Consider the sentence, "The speed limit in X-town is 50 kilometers an hour." A verification that this is, in fact, the current speed limit does not warrant the inference that any one driver is keeping his or her speed below this limit. The impact of the regulation is rather that the driver *ought* to keep his or her speed below that limit. The constraint representation makes this explicit: *If you are driving in X-town, your speed should be equal to or less than 50 km/h*. The relevance criterion is a conjunction of two simple propositions, that the person is driving and that he is in X-town. If he is riding a bicycle, speed limits for cars are irrelevant, and if he is not in X-town, then X-town traffic rules are also irrelevant. But when both conditions match the situation at hand, a speed greater than 50 km/h violates this regulation. In general, the constraint representation captures the intent behind norms.

Constraints vary along several dimensions. (a) The criteria R and S can be simple, consisting of elementary propositions, or complex, consisting of conjunctions of propositions. In principle, the conjunctions can be arbitrarily complex, but in practice, this is neither desirable nor needed. (b) In *state constraints*, the criteria only test for features of the current state of affairs, while in *path constraints*, they also test for features of the path -- the sequence of actions and outcomes -- by which the agent arrived at that state of affairs. (c) *Syntactic constraints* relate features of a state or a path to each other, while *semantic constraints* relate features of a state or path to some other entity, e.g., a solved example or some other authoritative source of information. In the context of filling out tax return forms, *if the sum in box n on page x is E, then the dollar amount in box m on page y should also be E* is a syntactic constraint; *if you deducted a work-related expense E last year, you cannot also deduct it this year* is a path constraint, and *the taxable income should be equal to what your employer reported it to be* is a semantic constraint.

A *constraint base* is any set of constraints. In the simplest case, this is an unordered set, without any higher-level structure. For some applications, it might be advantageous to impose a structure on the constraint base. It is useful to keep in mind that in the early days of production systems, several methods for structuring sets of production rules were proposed, but none of them became standard. It turned out that any advantage bought with such a technique could also be had by incorporating the relevant elements into the condition sides of the productions. Structured rule bases posed the problem of choosing which group of rules to execute at each moment in time. It may be that the same conclusion will ultimately emerge as the correct one with respect to constraint bases. At the present time, there is insufficient experience on which to base a firm opinion on this issue.

For most applications, it makes sense to associate a constraint base with a task or a particular environment. For example, the set of traffic laws for X-town is a constraint base for the task of moving about in that particular environment. Other more or less codified constraint bases include the rules of good manners, the APA format for literature

references and Robert's Rules of Order for holding meetings. Aesthetic judgments and the conventions of artistic genres – e.g., the Golden Rule of proportion and the rules of certain types of musical compositions such as a fugue or a concerto – are also constraints although many of them might be held implicitly.

A constraint base for a domain can be more or less complete. If a constraint base is complete, actions and sequences of actions vis-à-vis that task and their outcomes are correct if and only if they do not violate any of those constraints. The familiar definition of a linear ordering like the natural numbers is a small constraint base (more common formulations of the constraints in parentheses):

(i) *If E1 is an element, E1 is the first element, E2 is an element in the ordering, and E1 and E2 are distinct, then E2 cannot also be a first element.* (There is only one first element.)

(ii) *If E is an element in the ordering, then E has a successor.* (Every element has a successor.)

(iii) *If E1 is an element, E1 has successor E2, E3 is an element, and E2 and E3 are distinct, then E3 cannot be a successor to E1.* (The successor is unique.)

If an ordering of elements violates one or more of these constraints, it is not a linear ordering: There are either multiple initial elements, at least one element without a successor or an element with more than one successor. On the other side of the coin, this constraint base is complete. That is, any set of elements related to each other by the successor relation that does not violate any of these constraints is a bone fide linear ordering.

If a constraint base is incomplete, then there are solutions or performances that are incorrect or inappropriate in the relevant task environment, but which nevertheless do not violate any of the constraints. Completeness is obviously a matter of degree, and the consequences of a given degree of incompleteness depend on the application.

In principle, the criteria R and S can be encoded in any way a programmer finds convenient, e.g., as arbitrary Lisp functions. Working constraint-based systems have in fact implemented them as patterns, similar to the condition side of a production rule. This implementation assumes that the constraints are processed via pattern matching, an assumption that we will adhere to throughout this paper even though it is not the only way to implement this type of constraint. In the context of a pattern matching system, the *execution cycle* for a constraint base is as follows:

(i) Match the relevance criteria of all constraints against the current state of the world.
(ii) For each constraint with a satisfied relevance criterion, match their satisfaction conditions against the current state of the world.
(iii) Pass the constraint violations and satisfactions, if any, to a decision making module, learning mechanism or other process.
(iv) Wait until the state of the world changes due to actions by the performance mechanisms or to the dynamics of the environment itself.

(iv) Go to i.

A constraint base and an associated execution engine constitute a mechanism for passing judgment. A cognitive agent – human or machine – can use this mechanism to evaluate its own behavior. For example, a car driver can use the traffic norms to evaluate his or her own driving (*better slow down, I'm driving too fast, again*). The mechanism can just as easily be used to evaluate someone else's actions or their outcomes. For example, a police officer evaluates a person's driving by comparing it to the traffic regulations (y*ou were driving to fast, again*). In every day life, people are constantly performing both types of evaluations. On the one hand, they evaluate their own performance (*I was unsuccessful*; *that went really well*). On the other hand, they evaluate the performances of others (*that speaker was not very engaging*; *I really liked that movie*). Propositional representations do not support judgment and evaluative reasoning, because what is in fact the case has no bearing on what ought to be the case. The constraint representation encodes norms and other forms of evaluative knowledge directly.

The issue of the proper scope of the constraint representation is intricate. One would expect the division between constraints and propositions to be clear-cut. Traffic regulations, tax laws, aesthetic impulses and mathematical constructs are all man-made, so it stands to reason that they consist of norms. It seems equally obvious that pieces of knowledge that we traditionally regard as descriptive, e.g., the laws of nature as formulated in the natural sciences, cannot be cast as constraints. However, closer scrutiny reveals complexities. For example, consider the great conservation laws of chemistry and physics. These include the conservation of angular momentum and the conservation of mass in chemical reactions. How are these used in scientific practice? One chemist has this to say about the conservation of mass: "There is so much trust in the mass conservation law that an experimental result … that does not conform is automatically treated as false and rejected" (Gensler, 1987, p. 78). It appears that at least some natural laws operate as constraints in the sense that they constitute a basis for judging and evaluating the outcomes of scientific procedures, an intrusion of normative, constraint-based thinking into what we tend to regard as the stronghold of propositional knowledge and deductive inference. The issue arises how wide a swath of what we usually regard as descriptive knowledge can be reinterpreted as normative. This is an interesting question, but the usefulness of the constraint representation does not presuppose that all declarative knowledge can be represented in terms of constraints.

FIDELITY AND EFFICIENCY OF CBM

The Cognitive Fidelity of Constraint-Based Tutoring

Research on intelligent tutoring system was from its start intertwined with the psychological modeling of cognitive processes. For example, the early work by J. S. Brown, R. Burton and K. VanLehn produced not only BUGGY (Brown &Burton, 1978), a cognitive model of children's subtraction errors, but also DEBUGGY (Burton, 1982), a diagnostic system that could interpret incorrect answers to subtraction problem in terms of those errors, as well as Sierra and Repair Theory (Brown & VanLehn, 1980; VanLehn

1990), which together provide a formal explanation of how the incorrect subtraction procedures arise in the first place. Similarly, the intelligent tutoring systems that use the so-called *model-tracing technique* are claimed to be based on the ACT model of the human cognitive architecture (Anderson, 1993; Anderson et al., 1990; Anderson et al, 1995). Yet other types of ITS designs are claimed to be based on yet other cognitive theories (Doane & Sohn, 2000; Doane, Sohn & McNamara, 2000). Such connections to cognitive theories are often claimed as a source of strength for an approach or a particular system.

This type of claim invites scrutiny. An intelligent tutoring system is supposed to take over (some of) the tasks of a *teacher*. Why is it advantageous if its knowledge representations and computational processes mimic those of the *learner*? What type of cognitive fidelity matters for the design of an ITS? After all, the ITS interface is likely to deliver instruction in the form of more or less canned text, diagrams, video clips, solved examples and so on. The student never sees the knowledge structures that are internal to the system. Why does it matter whether the latter are isomorphic to the knowledge structures inside the learner's head?

For instruction to support learning, the learner's brain must be able to process and make use of the information contained in the instruction. Learning is a change in a knowledge representation, but representations cannot change in arbitrary ways. Efficient retrieval and application of knowledge requires a certain amount of organization, which in turn implies that changes are of specific types. A cognitive mechanism that computes a particular type of change is a *learning mechanism*. The statement that knowledge cannot change in arbitrary ways then translates into a claim that there is a limited number of learning mechanisms. The task of psychological research is to figure out what those mechanisms are; there are many proposals (association, discrimination, generalization, chunking, rule composition, etc.) but little by way of principles for choosing among them.

One plausible principle is that learning mechanisms differ with respect to the type of information they take as input. After all, an improvement in a cognitive skill, a change in its mental representation that provides a better fit to a task environment, cannot be achieved out of thin air. A learning mechanism adds structure to an existing skill, and hence must derive that structure from some source. We refer to this as the *Information Specificity Principle* (Ohlsson, in press-a). The general schema for a learning mechanism is therefore as illustrated in Figure 1.

Figure 1 in here

The implication for the design of instruction is that unless the type of information provided by instruction can serve as input to one or more learning mechanisms, then the information is inert and the instruction is likely to be ineffective. For example, suppose an instructor attempts to teach a concept by providing one example and one counterexample, in the hope that the differences will help the student sharpen his or her understanding of the target concept. This is equivalent to providing the learner with a set of alignable differences between instances and non-instances (Markman & Gentner, 1997). But suppose that the mind of the learner does not possess any discrimination mechanism; that is, there is no cognitive process in his or her head that takes a set of alignable differences as input and computes an improvement in the relevant concept representation. In that case, the instructional move fails. Similarly, if an instructor teaches by providing an analogy, and if it should turn out that people do not possess any cognitive mechanisms for constructing analogies, then that instructional tactic must fail.

In short, the Information Specificity Principle implies that instruction can only be effective if it provides information that can serve as input to one or more learning mechanisms in the student's head. This implies, in turn, that the designer of instruction has to have some hypothesis about those learning mechanisms and the types of information they take as input.

Learning mechanisms make implicit assumptions about how knowledge is represented. For example, a generalization mechanism might operate very differently if it is supposed to generalize propositions than if it is supposed to generalize production rules, and it might require different information as input to accomplish those two tasks. This is why the cognitive fidelity of the knowledge representation matters: It helps the system provide precisely the type of information that the learner is mentally equipped to learn from (according to some theory). This is the type of cognitive fidelity that matters for instruction and hence for ITS design.

In our own work on intelligent tutoring systems, we focused on the fact that the natural mode of instruction in an ITS is to provide help when the student is making errors; there is less reason for the system to intervene when the learner is producing correct answers and solutions. The focus of the theoretical effort was therefore directed towards a model of how students might detect and correct their errors, and how a tutor might help them do so.

The ability to catch one's own errors is paradoxical. If a person does not have enough knowledge to recognize an action or discourse as erroneous, then how can he or she catch the error? But if the person does have enough knowledge to recognize it as erroneous, then why is that action or discourse issued in the first place? The ability to catch one's errors forces a distinction between *generative* and *evaluative* knowledge (Norman, 1981; Ohlsson, 1996a). On the one hand, there must be a system (e.g., a set of strategic rules) for generating actions. General dispositions, heuristics, orienting attitudes and strategies acquired in past experience suffice to generate task relevant actions but they do not guarantee that those actions are correct (or else the task is not unfamiliar after all). The result is exploratory, tentative behavior (heuristic search; trial and error).When the relevant rule base is incomplete or incorrect, errors result. On the other hand, there must be a separate knowledge base for evaluating the (outcome of) an action and judging it as correct or incorrect, as the case might be. These two knowledge bases are independent in the sense that a piece of knowledge that appears in one does not

necessarily also appear in the other. The interaction between these two knowledge systems is what we subjectively experience as catching ourselves making an error.

This hypothesis only applies in scenarios in which the learner has at least some declarative knowledge of what correct solutions looks like, even before he or she is able to reliably generate them. This is in accord with intuition as well as instructional practice. Consider the task of recovering from being lost in unfamiliar surroundings. If the surroundings are truly unfamiliar, then one cannot tell, after walking for a while, whether one is closer or further from one's goal. Without ability to recognize the correct path, the walker cannot make an informed judgment about the path he or she is taking. Descriptive knowledge of what the desired solution looks like (*there is a gas station on the corner*) is crucial for catching oneself making errors and correcting wrong choices.

When an action outcome is found to be inappropriate, useless or incorrect (in a sense that depends on the task domain), the obvious response on the part of the learner is to revise the generative knowledge so as to avoid repeating that same error in the future. Over time, the information in the evaluative knowledge is gradually incorporated into the generative knowledge and the capability of the latter to generate correct actions gradually increases. The central process in skill acquisition is the migration of structure from the evaluator to the generator.

This hypothesis suggests that people possess a learning mechanism that can take information about a detected error, plus the current representation of the relevant skill, and compute a revision of that skill such that execution of that skill in future situations will not generate that same type of error again. In prior work, we have defined and implemented a specific machine learning algorithm, called *constraint-based rule specialization*, that can accomplish this (Ohlsson, 1993, 1996a; Ohlsson & Rees, 1991a, 1991b). The constraint-based specialization mechanism takes a constraint violation (and the current rule set) as input, and extracts condition elements from that information, which are then added to the condition side of the offending rule. The algorithm picks the new condition elements in such a way that future applications of the offending rule will either refuse to fire when the constraint is running a risk of becoming violated, or fire only when the constraint is already satisfied. The technical details of the learning algorithm have been published elsewhere, and the reader is referred to Ohlsson (1993, 1996a, in press-b) and Ohlsson & Rees, (1991a, 1991b).

For present purposes, the interesting question is what this learning mechanism implies with respect to instruction. If the mechanism takes a constraint violation as input, what does that mean? A constraint violation is a mismatch between a constraint of the domain and a particular state of affairs. For example, if a child counts five objects as, "one, two, three, three, four – four is the answer" this is obviously an error. The constraint is that in correct counting, every number word is to be used only once; the specific situation is that the number word "three" was used twice. The constraint violation therefore contains the information that the double use of "three" violates the one-one mapping constraint. If the child knows the one-one-mapping constraint, as some developmental psychologists have argued (Gelman & Gallistel, 1978), then he or she might be able to detect his or her error and react to the violation by constraining the procedure for generating the number words in the appropriate way.

The design of relevant instruction would conform to this pattern. An adult watching the child count might be tempted to say, for example, "you said 'three' two

times but you should only use each number just once." This type of message contains the two pieces: the constraint ("use each number just once") and the specific feature of the solution that violates that constraint ("you said 'three' two times"). In general, if the constraint-based specialization hypothesis is approximately true – that is, if people learn in some way that approximates or is computationally equivalent to this algorithm – then instruction that contains information about (a) the occurrence of an error, (b) the constraint that is violated, and (c) exactly how the current state of the problem violates that constraint ought to be effective. In short, the constraint representation exhibits cognitive fidelity in the sense that it provides the type of information that people can learn from (according to our learning theory).

The fidelity of the constraint representation can also be evaluated empirically, by observing students using constraint-based tutors. Several constraint-based tutors have been developed since 1996 at the Intelligent Computer Tutoring Group (ICTG[1]). These tutors have been evaluated in more than 30 studies in authentic classroom situations, with students enrolled in relevant courses. SQL-Tutor (Mitrovic, 1998; Mitrovic, 2003) was the first constraint-based system to be formally evaluated, with the results demonstrating that students using it achieved significant learning gains. After only two hours with SQL-Tutor, students outperformed their peers in the final examination, scoring an average of three quarters of a standard deviation higher on questions related to SQL query formulation (Mitrovic & Ohlsson, 1999). KERMIT, a database design tutor, has also been proven to be effective (Suraweera & Mitrovic, 2004). Students using this tutor during a regular laboratory session achieved significantly higher gains in their pre- to post-test scores ($t = 3.07$, $p < 0.01$) compared to their peers who received no feedback on their solutions apart for the ideal solution provided by the teacher. The effect size and power of this study were 0.63 and 0.75 respectively, at significance 0.05.

A second approach to empirical validation of cognitive fidelity of a knowledge representation is to investigate the shape of the resulting learning curves. Anderson (1993, especially pp. 32-35 and Chapters 7 and 8) have proposed that psychologically valid knowledge representations are indicated by smooth learning curves. When the unit of knowledge employed in the analysis of data corresponds to the unit of knowledge in the learner's head, empirical measures of student learning exhibit regular and smooth learning curves. The empirical data reported in (Mitrovic & Ohlsson, 1999; Mayo, 2001; Mayo & Mitrovic, 2001; Suraweera, 2001; Martin, 2002; Martin & Mitrovic, 2002; Mitrovic et al., 2002; Suraweera & Mitrovic, 2004; Mitrovic, 2005; Weerasinghe, 2005; Zakharov et al., 2005; Baghaei & Mitrovic, 2006; Baghaei et al., 2006) show that the constraint-based representation is doing well on this criterion. Figure 2 shows the learning curves for three constraint-based tutors. In these learning curves, we plotted student performance in terms of the average probability of violating a constraint as a function of the number of times the constraint has been applied. As the figure shows, when learning is measured in terms of probability of violating a particular constraint, the learning process is highly regular. The number of opportunities to violate a constraint, and hence to acquire the knowledge encoded in that constraint, is a strong determinant of the probability of further violations of that constraint.

[1] http://www.cosc.canterbury.ac.nz/tanja.mitrovic/ictg.html

Figure 2 in here

To further assess the cognitive fidelity of the constraint representation and its effects, we compared two versions of the database design tutor. In the first version, the tutoring messages were written on the basis of no other discipline than common sense. This often results in messages of the general form, *you did X here but you should have done Y*, a type of input for which learners might not have any learning mechanism, in which case instruction of this sort should have only weak effects. The second version of the tutoring system was identical, except that the tutoring messages were re-worded to conform closely to the implications of the constraint representation. That is, each message was worded to state the violated constraint in normal English, and to make clear exactly how the student's action violated that constraint. The results indicates that micro-engineering the content of the tutoring messages in this way provided an additional pedagogical benefit over and above that provided by the common sense based tutoring messages. The students who received theory-based feedback had a higher learning rate than their peers who received common-sense feedback, as can be seen in Figure 3.

Figure 3 in here

To summarize, the constraint-based approach to ITS design exhibits cognitive fidelity in both the theoretical and the empirical sense. The instruction it delivers is designed to serve as input into the learning mechanisms that is hypothesized in the constraint-based, rule specialization theory of learning from errors (Ohlsson, 1996a). The effectiveness of the instruction across multiple instructional topics, the extra pedagogical benefit of making tutoring messages closely conform to the implications of the theory and the resulting smooth learning curves all provide empirical support for the cognitive fidelity of the constraint representation.

The Efficiency of Constraint-Based Systems

Intelligent tutoring systems are complex artifacts, and their design and implementation requires efforts comparable to those of other artifacts such as a cell phones and airplanes. Large-scale and wide-spread deployment of tutoring technology presupposes production methods that are efficient in at least two senses: First, there are severe limits on how much resources in terms of time and manpower the design and implementation of an ITS for a given subject matter topic can consume. Second, the execution of the resulting software system should be robust and fast. These dimensions are additional criteria for the evaluation of different tutoring technologies in general and specifically the knowledge representations on which they are based.

Implementation efficiency. The key step in building a constraint-based system is to write the constraint base - that is, to capture what is true of the subject matter domain in a set of constraints. This is a knowledge acquisition task, as the latter term is used in the study of expert systems. As in other ITS technologies, it is necessary to consult someone who understands the target domain well, and his or her knowledge has to be made explicit and cast in the form of constraints. The more complete the constraint base, the better.

It is important that the constraints capture the principled aspects of the domain, and not merely surface features of particular training problems. For example, in an ITS designed to teach the task of deriving structural formulas from sum formulas in elementary chemistry, the constraints should capture the basic principles of the co-valent bond: Atoms strive towards the noble gas configuration, in which they have a full complement of electrons in their outer shell, and each co-valent bond requires two shared electrons. Sometimes particular facts have to be captured in constraints as well, e.g., that carbon atoms have four electrons in their outer shell. Constraints that capture the facts and principles of the domain constitute the core of the constraint base. Depending on the domain, it might be useful to also add constraints that refer to surface features of particular problems and their solutions, but this is not the main source of power of the constraint-based approach.

Three features of the constraint-based knowledge representation contribute to the efficiency of implementation. First, there is no need for a separate bug library, i.e., a representation of students' errors. The constraint base, although a representation of what is correct and true about the domain, also contains the relevant bug library. A bug – a mistake – will, by definition, violate one or more constraints of the domain. Hence, the set of possible errors that the students can make is specified once the constraint base is determined: The bug library is the set of all possible constraint violations. Specifying the constraint base is therefore also to implicitly specify the possible errors. For example, the chemistry constraint, *a covalent bond requires two electrons* implicitly specifies the two types of errors a student can make in specifying a single covalent bond: He or she can assign a single electron, and he or she can assign three or more electrons. Either action violates the constraint that each bond requires exactly two electrons.

This feature of the constraint-based approach means that the implementation of an ITS does not require empirical studies to identify the common student errors. This is a great advantage from a system building point of view. Such studies must collect large amounts of qualitative data and these have to be subject to painstaking analyses to identify and interpret errors. For example, the empirical data base for the original BUGGY/DEBUGGY system, including a large scale study of Nicaraguan school children and the so-called Southbay Study carried out in California, consisted of subtraction worksheets from more than two thousand students, gathered over a period of a couple of years (Brown & Burton, 1978; VanLehn, 1982). "…it requires a great deal of effort to build a database of bugs the size of the one the [sic] DEBUGGY now has. Lumping together the work of the six diagnosticians that have worked on the project over the last several years, we estimate that *four or five thousand hours* were spent" constructing the bug library (VanLehn, 1982, p. 38, italics in original).

The cost and effort of constructing a bug library would be worthwhile, if the task could be done once and for all. However, what little data are available on the issue

indicates that the repertoire of errors among students might depend on the way a subject matter is taught and hence do not transfer easily between classrooms or educational systems. There is some evidence that bug libraries do not transfer well between different populations of students (Payne & Squibb, 1990). This opens up the nightmare possibility that a new ITS has to be re-implemented, not only for every subject matter topic, but for every approach to teaching a subject matter and perhaps for every student population as well, threatening to make ITS development a non-cumulative and unsustainable enterprise. Systems built with the constraint-based approach, in contrast, require little revision when moved to a different instructional context, because the constraint base only encodes what is correct about the domain, an aspect of the subject matter that does not vary across instructional situations and contexts.

Computational efficiency. The process of executing a constraint base does not build chains of inferences, goal hierarchies, sequences of operators or other types of larger combinatorial structures. This is a great advantage, because the construction of larger structures is inevitably subject to combinatorial explosion, with the associated need for the system implementer to find some technique for pruning that explosion. In contrast, constraints are applied all at once, in parallel, and each constraint provides a signal (*relevant? satisfied? violated?*) that is independent of other constraints.

The modularity of the constraint base also facilitates debugging. A large proportion of the time and effort that goes into debugging a knowledge base is caused by unanticipated interactions among knowledge elements (inference rules, production rules, etc.), but in the constraint based approach there are no such interactions.

Because constraints are applied in parallel, execution of an ITS, once implemented, is fast and robust. In the standard implementation, constraints are implemented as patterns and applied via pattern matching. The latter is a well-studied problem, and several fast pattern matching algorithms exist, some of which exhibit a slower than linear growth with the number of patterns (Forgy, 1982).The fact that relevance patterns can be matched before satisfaction patterns and the matching of satisfaction patterns is only needed for relevant constraints also saves computational effort.

Summary

To summarize, the constraint-based approach to ITS design and implementation exhibits cognitive fidelity in the sense that it provides information that can serve as input into a well-specified learning mechanism and that mechanism is a plausible hypothesis about how human learn. Specifically, the constraint-based approach provides a basis for tutoring with the main pedagogical tactic that has been used in ITSs to date, namely to give feedback in response to errors. The learning gains produced by constraint-based systems, the slight but positive evidence that micro-engineering the feedback messages provides an additional increment of pedagogical power and the smooth learning curves that result when learning is plotted in terms of constraints provide empirical support for cognitive fidelity of the representation. Furthermore, the constraint-based approach is efficient in that (a) it requires no empirical studies of student errors and no explicit bug library, (b) the modularity of the constraint-base facilitates implementation and

debugging, and (c) the execution of a constraint-base requires no combinatorial processes and hence tends to be reliable and fast.

FIDELITY AND EFFICIENCY OF MODEL-TRACING

The knowledge representations that are commonly used in intelligent tutoring systems include production rules (Anderson, 1993; Anderson, Conrad & Corbett, 1993) and Bayesian networks (Mayo & Mitrovic, 2001; Mislevy & Gitomer, 1996; Conati, Gertner, & VanLehn, 2002). We compare the constraint-based representation to the first of these with respect to fidelity and efficiency, but space only allows a brief comment about the second.

Production rules are a format for representing cognitive skills. Since the pioneering work on problem solving skills by Newell and Simon (1972), production rules have become perhaps the most commonly adopted hypothesis regarding the mental representation of cognitive skills. It is a central component of many cognitive models (Ohlsson, in press-a). Analyses by Anderson (see Anderson, 1993, especially pp. 32-35 and Chapters 7 and 8), by Delaney, Reder, Staszeswki and Ritter (1998) and others have provided strong evidence that the production rule corresponds to a psychological unit of procedural knowledge. Semantic networks, propositional representations and schema representations also have impressive support in psychological research, but they represent declarative rather than practical or strategic knowledge and thus are not strictly speaking alternatives to production rules.

In a previous section, we argued that the type of cognitive fidelity that is of central importance for tutoring is that it specifies the type of information that can serve as input to one or more learning mechanisms that constitute plausible hypotheses about how people learn. To evaluate the cognitive fidelity of a tutoring technology, we must therefore ask on which hypotheses about learning it is based. There are many learning mechanisms associated with production rule models (Ohlsson, in press-a), but we will focus on those proposed in the context of the ACT model of the human cognitive architecture. The research group behind this model initiated an approach to the design and implementation of intelligent tutoring systems generally known as *model–tracing* (Anderson et al., 1990, 1995). They also claimed that such tutoring systems exhibit high cognitive fidelity because they are based on the ACT theory (Anderson et al, 1990).

Cognitive fidelity of model tracing tutors. Which learning mechanisms are hypothesized in the ACT model? This turns out to depend on the vintage. The ACT* model of the 1983 book, *The Architecture of Cognition*, sported five learning mechanisms: *strengthening/weakening* (of rules), *generalization*, *discrimination*, *rule composition* and *proceduralization*, the last two often being discussed in combination under the label *knowledge compilation* (Anderson, 1983). Of these, proceduralization can be conceptualized as a mechanism for turning task instructions into an initial set of (possible incomplete or erroneous) production rules, to be fine tuned by the other mechanisms.

The proceduralization mechanism implies that students can benefit from task instructions, and strengthening/weakening implies that students can benefit from both positive and negative feedback (*that's right* and *nope, that's wrong*), rather unproblematic claims. In the 1983 formulation of the ACT theory, the generalization and discrimination

mechanisms both depended on having a set of examples or instances available in memory – all positive ones in the case of generalization, and both positive and negative in the case of discrimination – and the instructional implications were complicated: The tutor would have to contrive to present the learner with practice problems such that there is a high probability that the learner generates the requisite set of instances to run either mechanism. Similarly, to feed the right type of information into the rule composition mechanism, the tutor would have to contrive a drill sequence that guarantees that the learner repeatedly executes certain rules in a fixed sequence. ("…leaving students to induce generalizations and discriminations from carefully juxtaposed examples…would have been the pedagogical implication of ACT*"; Anderson et al, 1990, p. 18). These implications pose serious difficulties in implementation: how would a tutoring system select practice problems to achieve the right inputs to the generalization, discrimination and composition mechanisms?

This question was never resolved, because the model-tracing tutors that were built in the period after the publication of the ACT* theory were not, in fact, designed in accordance with these instructional implications. These tutoring systems employed the standard instructional tactic of the ITS field: Wait until the student makes an error, and provide more or less extensive feedback in the form of an explanation. This instructional tactic does not follow from any of the five learning mechanisms of the ACT* theory. Neither strengthening, generalization, discrimination nor composition can take a current rule plus an explanation of an error as input and revise the rule accordingly. Weakening is a response to an error signal, but it does not take an explanation as input, merely the fact of a negative outcome, and its only effect is to lower the strength of a rule. So there is no role within that mechanism for the information provided by such an explanation.

Discrimination is also a response to an error (it requires at least one negative instance), but it changes the offending rule on the basis of a comparison with positive applications of the same rule, not on the basis of the information in an explanation. Proceduralization operates on verbal input, but that input consists of statements about the correct way to perform the task. (It is not very useful to turn errors into new production rules.) Also, proceduralization generates new rules, it does not revise old ones. In short, although the first generation of model-tracing tutoring systems were claimed to have high cognitive fidelity because they were "based on" the ACT* theory, there was in fact no connection between the learning mechanisms hypothesized in that theory and the mode of instruction that was implemented in those systems.

The five-mechanism theory of the ACT theory as of 1983 was abandoned. No critical analysis has been published to explain why. We have to assume that the theory did not mesh with the empirical findings of the ACT research group. In the next installment, ACT was equipped with an analogical learning mechanism that created new production rules in analogy with already learned ones. According to this version of the theory, analogical construction is the only way in which new production rules can enter the cognitive system; there are no other learning mechanisms: "…the only way new procedural knowledge, in the form of production rules, enters the system is by the process of analogizing to some previous declarative knowledge" (Blessing & Anderson, 2002, p. 607).

What are the instructional implications of this assumption? It predicts, strictly speaking, that tutoring cannot work. That is, there is no mechanism in the learner's head

that can take a feedback message and a current skill as inputs and revise that skill according to the content of that message. The only way to instruct according to the analogy-based version of the ACT theory is to carefully sequence practice problems in such a way that when the learner comes to a problem for which he or she needs a particular production rule, a prior analogue to that rule is available in memory. We know of no model-tracing tutoring system that instructs solely through clever problem selection and without feedback messages.

The rule analogy hypothesis, as originally stated, was also abandoned. Once again, the exact mismatches with data that drove this revision of the theory have not been published. Analogy was later re-conceptualized as involving two mechanisms, a basic ability to notice similarities between objects that is assumed to be part of the cognitive architecture and a strategy of constructing analogies in response to impasses or difficulties; the latter is assumed to be learned and hence can operate differently in different domains (Salvucci & Anderson, 2001). The instructional implication of this version is the same as for the preceding version: If analogy is the only learning mechanism, then people cannot learn from tutorial feedback and pedagogy is reduced to clever sequencing of practice problems.

ACT* eventually morphed into ACT-R through the marriage between the symbolic architecture and an extensive apparatus for sub-symbolic, quantitative learning known as the rational analysis of memory and decision making. The version of the ACT-R theory published in *The atomic components of thought* (Anderson & Lebiere, 1998), did not specify any mechanism for the construction of new rules. No learning mechanism, no implications for instruction or ITS design.

The latest version of the ACT-R theory is once again equipped with a single rule learning mechanism, called *rule compilation* (Taatgen & Anderson, 2002; Taatgen & Lee, 2003). This mechanism is a descendant of the rule composition mechanism of 1983 vintage. Like its predecessor, the new mechanism operates by combining two rules that repeatedly execute in temporal sequence into a single rule. In the course of the combination, it eliminates retrievals from declarative memory by specializing the rule: The retrieved facts are incorporated as condition elements into the rule, thus creating a single, task specific rule that does not need to retrieve elements from declarative long-term memory in order to execute. Once again, claims are made by members of the ACT research group that this is the only symbolic learning mechanism. ("New production rules are learned through production compilation." Taatgen & Lee, 2003, p. 67).

What are the instructional implications of this mechanism? Its input consists of two rules, their instantiations and the information that they fired in chronological sequence. The instructional implication is, once again, that an instructor can only facilitate the operation of this learning mechanism by choosing practice problems in such a way that they will cause the relevant rules to execute in the sequence required for production compilation to generate the rules of the target skill. Once again, there is no role for instruction in the sense of tutorial discourse, e.g. feedback messages and explanations.

Model-tracing tutors share with the ACT theory the use of production rules as a representation for cognitive skills, but there is no other sense in which they are "based on" that theory. However, the production rule format is not unique to model-tracing tutors. Constraint-based tutors also assume that cognitive skills are encoded in production

rules; the purpose of the constraint-based rule specialization mechanism is to revise production rules. The evidence for the cognitive fidelity of the production rule format thus supports both tutoring technologies equally.

In short, we find that the claim that model-tracing tutors are "based on" the ACT theory of the human cognitive architecture to be spurious. The type of cognitive fidelity that matters for ITS design – that the instruction delivered is the type of information that can serve as input to one or more of the learning mechanisms in the learner's head, as specified in some plausible theory – is entirely absent from this line of tutoring systems. No version of the ACT theory ever contained a learning mechanism that can learn from feedback in response to an error, i.e., from the primary form of instruction delivered by the model-tracing tutors that have been published to date. Those tutoring systems lack cognitive fidelity in the sense that matters.

Efficiency of model-tracing. Model-tracing tutors operate, in principle[2], by matching production rules to students' problem solving steps. The rule base contains both correct and buggy rules. If a student step matches a correct rule, there might not be any reason to intervene. If the step matches a buggy rule, the match can be passed to an instructional decision maker that decides what instruction, if any, to deliver. In practice, the simplest pedagogy is to associate instruction in the form of canned messages directly with the corresponding buggy rules.

(a) Efficiency of implementing an expert model. To implement a model-tracing system requires an explicit encoding of the correct or desired target skill, a so-called expert or ideal student model. Without such a model, the model-tracing system cannot know whether a student step that does not match any buggy rule is correct or marred by a bug not represented by any buggy rule. Constraint-based tutors also require an encoding of the correct subject matter. In the former case, this model is a rule base, in the latter, a constraint-base. One aspect of efficiency is thus whether it is easier or more difficult to construct one type of knowledge base than the other.

One way to answer this question is to compare the implementation time per unit of knowledge. Anderson et al. (1995) estimate the time necessary for acquiring a single production rule to be 10 hours on average, while the reported time per constraint (Mitrovic & Ohlsson, 1999) is considerably shorter (1.1 hours). Comparative studies of efficiency are rare. Mitrovic, Koedinger and Martin (2003) reported a limited comparison in which a small portion of KERMIT, a constraint-based tutor for database design, was re-implemented using the model-tracing approach. The model-tracing version required 25 production rules plus additional problem-specific information, represented in form of chunks, but covered only a small number of problems. On the other hand, the equivalent part of the constraint-based version of KERMIT included 23 constraints, required no extra problem-specific information other than the ideal solution for each problem, and was problem independent, enabling any number of problems to be taught. Kodaganallur, Weitz and Rosenthal (2005) built one model-tracing and one constraint-based tutoring system for one and the same task domain, a simple form of statistical hypothesis testing.

[2] We recognize that model-tracing is not a static concept and that there might be different ways to implement this technique. For brevity, we focus on the canonical version, as it is described in publications of the model-tracing systems, e. g., Anderson et. al (1990, 1995).

They reported that the rule base, containing 76 rules, required over four person-months, i.e., approximately 120 person days, and that the constraint-base, comprising 43 constraints, required 23 person-days (an average of two hours per constraint). What little quantitative data are available thus suggest that an expert model in the form of a constraint base requires considerable less work than the corresponding rule-based expert model.

(b) Efficiency of implementing a buggy model. To implement a model-tracing tutor requires a buggy model, i.e., a rule base that encodes the common errors that students make on the path to mastery. The buggy rules can only be found through empirical studies. Such studies require the collection of large amounts of qualitative data which have to undergo painstaking analysis. The errors have to be understood in enough depth so that they can be encoded in terms of production rules.

In contrast, constraint-based tutoring systems do not need a separate buggy model. The set of constraints implicitly defines the set of buggy solutions. In a constraint-based system, a bug is a step, or sequence of steps, that violates one or more constraints. The universe of bugs is thus the set of all possible ways to violate the constraints. The specification of this universe requires no additional effort, over and above implementing the expert module. The universe of errors is given when the constraint-base is complete, or as complete as it is practical to make it in any given system.

The savings in efficiency of implementation caused by this difference is difficult to overestimate. The empirical studies that underpinned the buggy models of the first generation tutoring systems were extremely time consuming and labor intensive; they sometimes took several man-years. Furthermore, they place the developer of intelligent tutoring systems in the uncomfortable position of having to pose as a cognitive psychologist and conduct empirical studies and analyze empirical data. Not all research groups that aim to develop ITSs possess the expertise or the inclination for this type of work. The constraint-based approach does away with the need for such studies. In short, when it comes to efficiency of implementation, the constraint-based approach is vastly superior to the model-tracing approach.

(c) Efficiency of execution. Both constraint-based and rule-based tutoring systems need to contain a pattern matcher and operates primarily via pattern matching. This is advantageous because pattern matching is computationally cheap. In their basic forms, the two approaches are equal on this variable.

However, the model-tracing approach potentially suffers from the following problem: Consider a solution path in which the (overt) action A_i is followed by some other (overt) action A_j. In the straightforward case, these actions can be traced by matching them to the production rules

R': Goal, C1, C2, C3,…, Ck ===> $A_{i,}$

and

R'': Goal, D1, D2, D3,…,Dn ===> A_j

where C1, C2, C3,…, Ck and D1, D2, D3,…,Dn are condition elements (situation features).

A problem arises if the student has to engage in extensive thinking between Ai and Aj. That is, if his mental solution path is denser than the overt solution path, so that the better model of the student's path is:

A_i, op_1, op_2, op_3, …, op_n, A_j.

where op_1, op_2, op_3, …, op_n, are cognitive, internal operations, inferences, subgoaling steps, and so on.

The problem resides in the fact that these internal operations change the content of working memory and hence affect the match of rule R". Unless the working memory content is updated with the changes caused by the internal, intermediate operations, R" might fail to match, but for the wrong reason, and the overt step A_j might be interpreted as incorrect even though it is correct (if R" is a correct rule) or vice versa (if R" is a buggy rule). In order to decide whether A_j was the correct step (i.e.., whether it matches any correct production rule), the tutoring system has to know the state of working memory at the moment when the decision to execute A_j is taken - that is, the state of working memory at the end of the sequence of internal operations op_1, op_2, op_3, …, op_n. The implication is that the tutoring system must contain rules that control these operations as well.

But how is the system to trace the operations op_1, op_2, op_3, …, op_n? By definition, they are not overt and hence leave no trace in the interface. The system could try to search forward from the last overt action A_i and see whether it can find a path to A_j, using only the correct rules. If so, it is reasonable to assume that A_j is a correct step. However, the model-tracing tutor now has to search. In the words of the inventors of model-tracing: "Often it is not clear which of a number of solution paths a student is on and the production system has to be used nondeterministically to enable a number of paths to be traced until disambiguating information is encountered." (Anderson et al., 1990, p. 35). Search is always combinatorial and hence potentially explosive in terms of the number of possible paths. A constraint-based tutor does not suffer from this problem, because it does not need to trace each individual action.

We note that this problem can be finessed by designing the interface in such way that the students' overt actions are very fine grained and hence any sequence of internal cognitive operations between successive overt actions is likely to be short. This appears to have been the case with the Lisp Tutor (Anderson, Conrad & Corbett, 1989). This tutoring system apparently traced each individual key stroke during coding. Whether it is pedagogically desirable to give feedback at this fine granularity is an open question.

In short, in the basic mode, constraint-based and rule-based tutors are comparable with respect to efficiency of execution. Both can operate their basic functions with a pattern matcher. To the extent that a model-tracing tutor needs to interpolate cognitive steps between overt, observable steps, it runs the risk of combinatorial explosion. There is no corresponding problem in the constraint-based approach.

CONCLUSION

There are two broad classes of dimensions that affect the usefulness of a knowledge representation for the purpose of intelligent tutoring. The first are aspects of cognitive fidelity, how closely the knowledge representation and its associated processes correspond to the knowledge representations and processes in the head of the learner. Within this broad concept, we can distinguish between two dimensions of fidelity. The first dimension is the cognitive fidelity of the representation itself, i.e., the extent to which a unit of knowledge in the representation corresponds to the units of knowledge in the student's mind. Along this dimension, production rules, as a representation for procedural knowledge, have the edge over all others, given the repeated demonstrations that analyzing learning data rule by rule gives more regular results than analyses in terms of number of training problems. However, constraints and propositions, as representations of normative and declarative knowledge are also plausible and supported by empirical data. When we analyze learning data in terms of constraints, we encounter the same regular pattern as when data are analyzed in terms of rules.

A second dimension of cognitive fidelity concerns the learning mechanism that is supposed to operate on the knowledge representation. We do not know what learning mechanisms the mind contains, but hypotheses about learning mechanisms can be evaluated on the basis of plausibility, generality, sufficiency, adequacy with respect to such data as might be available, and so on. To apply this criterion, there has to be a well-specified learning mechanism with specific implications for instruction. We find that the succession of learning mechanisms proposed in the context of the ACT theory, the theory that is claimed to underpin model-tracing tutoring systems, shows no sign of converging over time and appears unrelated to the main pedagogical tactic employed in model-tracing tutors (give feedback when the student makes an error). On this dimension, the constraint-based approach is superior. The constraint-based mechanism for learning from error that is described in Ohlsson (1996a) is psychologically plausible, demonstrably general (Ohlsson, 1993) and sufficient to learn ecologically valid skills (Ohlsson & Rees, 1991a; Ohlsson, Ernst & Rees, 1991). It is directly tied to the constraint representation and generates specific implications about which type of instruction that ought to be effective. The empirical data support these implications.

The second cluster of dimensions concerns efficiency. It matters how easy or difficult a particular representation and its associated design philosophy are to use, and how fast the finished tutor can execute. With respect to efficiency of implementation, the constraint-based approach is superior to model-tracing. The reason is that building a constraint-base does not require empirical studies to identify common student errors, bugs or misconceptions. The relevant universe of errors is specified indirectly, as the set of possible constraint violations, once the correct subject matter has been specified and cast as constraints. The auxiliary cause of the superior implementation efficiency of the constraint-based approach is the modularity of the constraint base. Constraints are applied in parallel and they do not require any combinatorial processes, so debugging a constraint base is comparatively easy. Constraints do not combine into larger structures, so there is no risk for combinatorial explosion. The importance of these advantages in the ease of implementation cannot be emphasized enough.

Designing a knowledge representation to support intelligent tutoring by machine is a complex enterprise that needs to be evaluated along multiple dimensions. We should not expect any one representation to be superior to all competitors in every respect. Instead, each representation has its own unique profile of advantages and disadvantages, and the choice of representation might depend on the interaction between the advantages profile and the demands of the particular application. Table 1 summarizes the relative advantages we found in the course of our analysis.

Table 1 in here

A particular knowledge representation cannot be expected to be superior to all others in every respect. More reasonable expectations are that every knowledge representation will exhibit a profile of advantages and disadvantages for ITS work, and that any pair of representations will turn out to be close on some dimensions and differ along others. For example, if we compared either model tracing or constraint-based modeling to the Bayesian network representation, we would expect to find a different pattern of similarities and differences, advantages and disadvantages, compared to those in Table 1. For example, unlike production rules and constraints, the cognitive fidelity of Bayes Rule is questionable because people are not fully rational when reasoning with probabilities and often make smaller changes than Bayes Rule dictate. Implementation efficiency is also an issue, because to use Bayesian reasoning in a serious way, the system implementer has to have some prior conditional probabilities to hang on the links in the network, and securing those requires work. On the other hand, Bayesian networks win hands down over symbolic representations on the computational efficiency dimension, because number crunching is fast and reliable and the quantitative nature of the information provides for graded responses, graceful degradation and soft landings, all desirable properties in an ITS. A particular profile of strengths and weaknesses will interact with purpose and intended use of a system, resources available for building it, and other factors. The field might benefit from the emergence of a set of standard dimensions on which to evaluate current and future knowledge representations for ITS.

Future work in the ITS field is likely to advance along a dimension not mentioned here: To what extent does the tutoring system support the full range of ways in which people learn? It is highly unlikely that human beings learn from a single learning mechanism. Instead, it is plausible that the mind has evolved a repertoire of learning mechanisms that are optimized to make use of the different sources of information available in the environment. The latter include negative feedback, but also positive feedback, task instructions, regularities in the execution trace such as temporal successions, impasses, search results, contrasts between negative and positive cases, and much more (Ohlsson, in press-a). According to the Information Specificity Principle, each learning mechanism provides an opportunity to support learning by providing information that can serve as input into that learning mechanism. Presumably, a tutoring system that feeds the relevant types of information into a wider range of mechanisms will provide more support for learning than systems that specialize in supporting a particular mechanism.

To date, the main instructional strategy of most ITSs has been to intervene when a student makes an error. The constraint-based approach was formulated with this particular form of instruction in mind. However, even a casual inspection of transcripts of interactions between human tutors and students reveal a multitude of types of tutoring moves, including an abundance of positive feedback. In future work, we intend to extend our systems to deliver other forms of instruction as well. It is an open question whether the constraint representation is sufficient to support those additional forms of instruction or whether additional knowledge representations will be needed. The future might belong to hybrid systems that encompass multiple knowledge representations that support a variety of tutoring tactics.

REFERENCES

Anderson, J. R.: 1983, 'The architecture of cognition'. Cambridge, MA: Harvard University Press.

Anderson, J. R.: 1993, 'Rules of the mind'. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Boyle, C. F., Corbett, A. T. and M.W. Lewis: 1990, 'Cognitive modeling and intelligent tutoring'. *Artificial Intelligence*, 42, 7-49.

Anderson, J. R., Conrad, F. G. and A. T. Corbett: 1989, 'Skill acquisition and the LISP Tutor'. *Cognitive Science*, 13, 467-505.

Anderson, J. R, Corbett, A. T., Koedinger, K. R. and R. Pelletier: 1995, 'Cognitive tutors: Lessons learned'. *Journal of the Learning Sciences*, *4*(2), 167-207.

Anderson, J. R. and C. Lebiere: 1998, 'The atomic components of thought'. Mahwah, NJ: Erlbaum.

Baghaei, N. and A. Mitrovic: 2006, 'A Constraint-based Collaborative Environment for Learning UML Class Diagrams'. In M. Ikeda, K. Ashley, and T.-W. Chan (eds.): *Proc. 8th Int. Conf. on Intelligent Tutoring Systems,* 176 – 186.

Baghaei, N., Mitrovic, A. and W. Irvin: 2006, 'Problem-Solving Support in a Constraint-based Intelligent Tutoring System for UML'. *Technology, Instruction, Cognition and Learning*, 4(2), 113-137.

Blessing, S. B. and Anderson, J. R.: 1996, 'How people learn to skip steps'. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 22, 576-598. [Reprinted in T. A. Polk & C. M. Seifert (Eds.), (2002). Cognitive modeling (pp. 577-620). Cambridge, MA: MIT Press.]

Brown, J. S. and R.R. Burton: 1978, 'Diagnostic models for procedural bugs in basic mathematical skills'. *Cognitive Science*, 2, 155-192.

Brown, J. S., & VanLehn, K.: 1980, 'Repair theory: A generative theory of bugs in procedural skills'. *Cognitive Science*, *4*, 379-426.

Burton, R.: 1982, 'Diagnosing bugs in a simple procedural skill'. In D. Sleeman & J. S. Brown (eds.), *Intelligent tutoring systems* (pp. 17-183). New York, NY: Academic Press.

Conati, C., Gertner, A. and K. VanLehn: 2002, 'Using Bayesian networks to manage uncertainty in student modeling'. *User Modeling and User-Adapted Instruction*, 12, 371-417.

Delaney, P. F., Reder, L. M., Staszewski, J. J. and Ritter, F. E.: 1998, 'The strategy-specific nature of improvement: The Power Law applies by strategy within task'. *Psychological Science*, 9(1), 1-7.

Doane, S. M. and Y. W. Sohn: 2000, 'ADAPT: A predictive cognitive model of user visual attention and action planning'. *User Modeling and User-Adapted Interaction*, *10*, 1- 45.

Doane, S. M., Sohn, Y. W., McNamara, D. S. and D. Adams: 2000, 'Comprehension-based skill acquisition'. *Cognitive Science*, *24*, 1-52.

Forgy, C. L.: 1982, 'Rete: A fast algorithm for the many pattern/many object pattern match problem'. *Artificial Intelligence, 19*, 17-37.

Freuder, E. and A. Mackworth (eds.): 1994, 'Constraint-based reasoning'. Cambridge, MA: MIT Press.

Gelman, R. and C. R. Gallistel: 1978, 'The child's understanding of number'. Cambridge, MA: Harvard University Press.

Gensler, W. J.: 1987, 'Impossibilities in chemistry: Their rise, nature, and some great falls'. In P. J. Davis and D. Park (Eds.), *No way: The nature of the impossible* (pp. 73-89). New York, NY: Freeman.

Gentner A., Conati C. and K. VanLehn: 1998, 'Procedural help in Andes: Generating hints using a Bayesian network student model.' In *Proc. 15th National Conf. Artificial Intelligence (AAAI-98)*, MIT Press, 106-111.

Glaser, R.: 1967, 'Some implications of previous work on learning and individual differences'. In R. Gagne (Ed.), *Learning and individual differences* (pp. 1-18). Columbus, OH: Merrill.

Hartley, D. and A. Mitrovic: 2002, 'Supporting learning by opening the student model'. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) *Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002*, Biarritz, France, LCNS 2363, 2002: 453-462.

Hawkes, L. W. and S.J. Derry: 1989/90, 'Error diagnosis and fuzzy reasoning techniques for intelligent tutoring systems'. *Journal of Artificial Intelligence in Education*, 1, 43-56.

Kodaganallur, V., Weitz, R. R. and D. Rosenthal: 2005, 'A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms'. *Int. J. Artificial Intelligence in Education,* 15, 117-144.

Koedinger, K. R., Anderson, J. R., Hadley, W. H. and M.A. Mark: 1997, 'Intelligent Tutoring Goes to School in the Big City'. *Int. J. Artificial Intelligence in Education*, 8, 30-43.

Lehler, W.: 1988, 'Constraint programming languages'. Reading, MA: Addison-Wesley.

Markman, A. B.: 1999, 'Knowledge representation'. Mahwah, NJ: Erlbaum.

Markman, A. B. and D. Gentner: 1997, 'The effects of alignability on memory'. *Psychological Science*, 8, 363-367.

Martin, B.: 2002, 'Intelligent Tutoring Systems: The practical implementation of constraint-based modelling'. PhD Thesis, University of Canterbury.

Martin, B. and A. Mitrovic: 2000, 'Tailoring Feedback by Correcting Student Answers'. In: G. Gauthier, C. Frasson and K. VanLehn (eds), *Proc. ITS'2000*, Springer, pp. 383-392.

Martin, B. and A. Mitrovic: 2002, 'Automatic Problem Generation in Constraint-Based Tutors'. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) *Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002*, Biarritz, France, LCNS 2363, pp. 388-398.

Martin, B. and A. Mitrovic: 2003, 'Domain Modeling: Art or Science?' In: U. Hoppe, F. Verdejo & J. Kay (ed) *Proc. 11th Int. Conference on Artificial Intelligence in Education*, IOS Press, pp. 183-190.

Martin, J. and K. VanLehn: 1993, 'OLAE: Progress toward a multi-activity, Baysian student modeler'. In S. Brna, S. Ohlsson & H. Pain (Eds.), *Artificial intelligence in education: Proceedings of AIED93,* Charlottesville, VA: AACE, pp. 410-417.

Mayo, M.: 2001, 'Bayesian Student Modeling and Decision-Theoretic Selection of Tutorial Actions in Intelligent Tutoring Systems', PhD Thesis, University of Canterbury.

Mayo, M. and A. Mitrovic: 2001, 'Optimizing ITS Behaviour with Bayesian Networks and Decision Theory'. *International Journal on Artificial Intelligence in Education*, 12(2), 124-153.

Miguel, I.: 2004, 'Dynamic flexible constraint satisfaction and its application to AI planning'. Berlin, Germany: Springer-Verlag.

Mislevy, R. J. and D. H. Gitomer: 1996, 'The role of probability-based inference in an intelligent tutoring system'. *User-Modeling and User-Adapted Instruction*, 5, 253-282.

Mitrovic, A.: 1997, 'SQL-Tutor: a preliminary report' (Technical Report No. TR-COSC 08.97). Christchurch, New Zealand: Computer Science Department, University of Canterbury.

Mitrovic, A.: 2001, 'Self-assessment: how good are students at it?' In: J. Gilbert, R. Hubscher, S. Puntambekar (eds) *Proc. AIED 2001 Workshop on Assessment Methods in Web-Based Learning Environments & Adaptive Hypermedia*, San Antonio, 2-8.

Mitrovic, A.: 2002, 'NORMIT, a Web-enabled tutor for database normalization'. In Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson & C.-H. Lee (Eds.) *Proceedings of the Int. Conf. on Computers in Education, ICCE 2002*, Los Alamitos, CA: IEEE Computer Society, pp. 1276-1280.

Mitrovic, A.: 2003, 'Supporting Self-Explanation in a Data Normalization Tutor'. In: V. Aleven, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo, K. Yacef (eds) *Supplementary proceedings, AIED 2003*, pp. 565-577.

Mitrovic, A., Koedinger, K. and B. Martin: 2003, 'A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling'. In: P. Brusilovsky, A. Corbett, F. de Rosis (eds) *Proc. 9th Int. Conf. User Modeling*, Springer-Verlag, LNAI 2702, pp. 313-322.

Mitrovic, A. 2005, 'The Effect of Explaining on Learning: a Case Study with a Data Normalization Tutor'. In: C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (eds) *Proc. Artificial Intelligence in Education*, IOS Press, pp. 499-506.

Mitrovic, A. and B. Martin: 2002, 'Evaluating the effects of open student models on learning'. In: P. de Bra, P. Brusilovsky and R. Conejo (eds) *Proc. 2nd Int. Conf on Adaptive Hypermedia and Adaptive Web-based Systems AH 2002*, Malaga Spain, LCNS 2347, 296-305.

Mitrovic, A., Martin, B. and M. Mayo: 2002, 'Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor'. Int. J. *User Modeling and User-Adapted Interaction*, 12(2-3), 243-279.

Mitrovic, A. and S. Ohlsson: 1999, 'Evaluation of a Constraint-Based Tutor for a Database Language'. *Int. J. Artificial Intelligence in Education*, 10(3-4), 238-256.

Norman, D.: 1981, 'Categorization of action slips'. *Psychological Review, 88*, 1-15.

Ohlsson, S.: 1986, 'Some principles of intelligent tutoring'. *Instructional Science, 14*, 293-326.

Ohlsson, S.: 1991, 'System hacking meets learning theory: Reflections on the goals and standards of research in Artificial Intelligence and education'. *Journal of Artificial Intelligence in Education,* 2(3), 5-18.

Ohlsson, S.: 1992, 'Constraint-based student modeling'. *Journal of Artificial Intelligence and Education*, 3(4), 429-447.

Ohlsson, S.: 1993, 'The interaction between knowledge and practice in the acquisition of cognitive skills'. In A. Meyrowitz and S. Chipman (Eds.), *Foundations of knowledge acquisition: Cognitive models of complex learning* (pp. 147-208). Norwell, MA: Kluwer.

Ohlsson, S.: 1996a, 'Learning from performance errors'. *Psychological Review*, 103, pp. 241-262.

Ohlsson, S.: 1996b, 'Learning from error and the design of task environments'. *Int. Journal of Educational Research*, *25*(5), 419-448.

Ohlsson, S.: (in press-a), 'Computational models of skill acquisition'. In R. Sun (Ed.), *The Cambridge handbook of computational modeling*. Cambridge, UK: Cambridge University Press.

Ohlsson, S.: (in press-b), 'Order effects in constraint-based skill acquisition'. In F. E. Ritter, J. Nerb, T. O'Shea and E. Lehtinen (Eds.), *In order to learn: How the sequence of topics influence learning*. New York, NY: Oxford University Press.

Ohlsson, S., Ernst, A. and E. Rees: 1992, 'The cognitive complexity of doing and learning arithmetic'. *Journal for Research in Mathematics Education*, 23(5), 441-467.

Ohlsson, S. and E. Rees: 1991a, 'The function of conceptual understanding in the learning of arithmetic procedures'. *Cognition & Instruction*, *8*, 103-179.

Ohlsson, S. and E. Rees: 1991b, 'Adaptive search through constraint violations'. *Journal of Experimental and Theoretical Artificial Intelligence, 3,* 33-42.

Payne, S. and H. Squibb: 1990, 'Algebra mal-rules and cognitive accounts of errors'. *Cognitive Science*, *14*, 445-481.

Reason, J. T.: 1992, 'Cognitive underspecification: Its variety and consequences'. In B. J. Baars, (Ed.), *Experimental slips and human error: Exploring the architecture of volition* (pp. 71-91). New York, NY: Plenum Press.

Salvucci, D. D. and J. R. Anderson: (2001), 'Integrating analogical mapping and general problem solving: the path-mapping theory'. *Cognitive Science*, *25*, 67-110.

Self, J. A.: 1990, 'Bypassing the intractable problem of student modeling'. In C. Frasson & G. Gauthier (Eds.), *Intelligent tutoring systems: At the crossroads of artificial intelligence and education* (pp. 107-123). Norwood, NJ: Ablex.

Sleeman, D., and J. S. Brown, (Eds.): 1982, 'Intelligent tutoring systems'. New York, NY: Academic Press.

Sleeman, D., Hirsch, H., Ellery, I. and I-Y. Kim: 1990, 'Extending domain theories: Two case studies in student modeling'. *Machine Learning, 5*, 11-37.

Sleeman, D., Kelly, A. E., Martinak, R., Ward, R. D. and J.L. Moore: 1989, 'Studies of diagnosis and remediation with high school algebra students'. *Cognitive Science, 13*, 551-568.

Soloway, E. and J. Spohrer: 1989, 'Studying the novice programmer'. Hillsdale, NJ: Erlbaum.

Suraweera, P. and A. Mitrovic: 2004, 'An Intelligent Tutoring System for Entity Relationship Modelling'. *Artificial Intelligence in Education*, 14(3-4), 375-417.

Taatgen, N. A. and J. R. Anderson: 2002, 'Why do children learn to say "Broke"? A model of learning the past tense without feedback'. *Cognition*, 86, 123-155.

Taatgen, N. A. and F. J. Lee: 2003, 'Production compilation: A simple mechanism to model complex skill acquisition'. *Human Factors*, *45*, 61-76.

VanLehn, K.: 1982, 'Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills'. *Journal of Mathematical Behavior*, 3(2), 3-71.

VanLehn, K.: 1990, 'Mind bugs: The origins of procedural misconceptions'. Cambridge, MA: MIT Press.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A. and M. Wintersgill: 2005, 'The Andes Physics Tutoring System: Lessons Learned'. *Artificial Intelligence in Education*, 15(3), 147-204.

Weerasinghe, A.: 2003, 'Exploring the effects of self-explanation in the context of a database design tutor'. MSc thesis, University of Canterbury.

Weerasinghe, A. and A. Mitrovic: 2005, 'Facilitating Deep Learning through Self-Explanation in an Open-ended Domain'. *Int. J. of Knowledge-based and Intelligent Engineering Systems*, IOS Press, 9, 1-17.

Zakharov, K., Mitrovic, A. and S. Ohlsson: 2005, 'Feedback Micro-engineering in EER-Tutor'. In: C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (eds) *Proc. Artificial Intelligence in Education*, IOS Press, pp. 718-725.

Table headings

Table 1. Summary of Comparison Between Constraint-Based and Rule-Based Tutoring.

Table 1. Summary of Comparison Between Constraint-Based and Rule-Based Tutoring.

Dimension	Constraint-Based	Rule-Based
Fidelity		
Knowledge representation	Supported	Well supported
Mechanism for learning from negative feedback	Strongly supported	Non-existent
Efficiency		
Implementation efficiency	Does not require a bug library	Requires a bug library
	Modularity of constraints facilitates debugging	Rule interactions complicate debugging
Computational efficiency	Modularity avoids combinatorial explosion	Combinatorial explosion for interpolated steps
	Fast execution through pattern matching	Fast execution through pattern matching

Figure captions

Figure 1. Schematic for learning mechanisms.

Figure 2. Learning curves from evaluations of NORMIT, KERMIT and SQL-Tutor.

Figure 3. Learning curves for students receiving theory-based versus intuitive feedback.

Figure 1

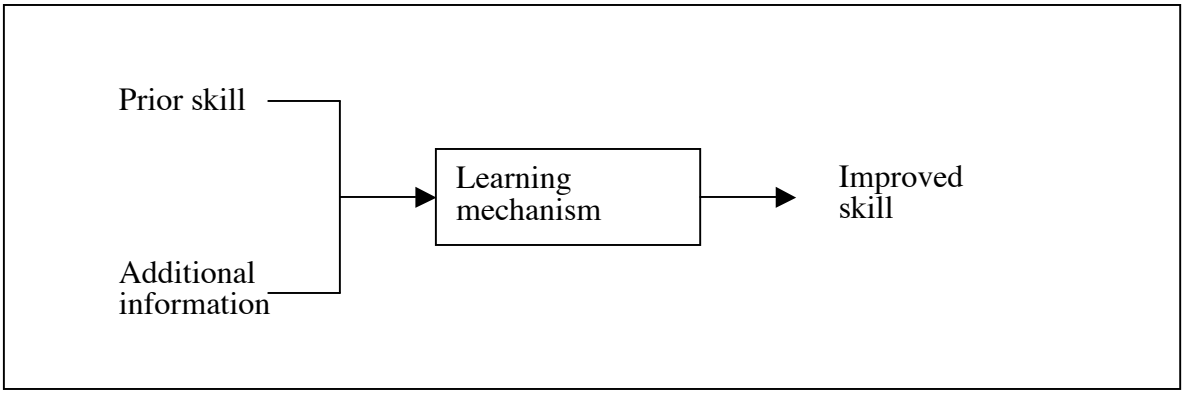

Prior skill
Additional information
Learning mechanism
Improved skill

Figure 2

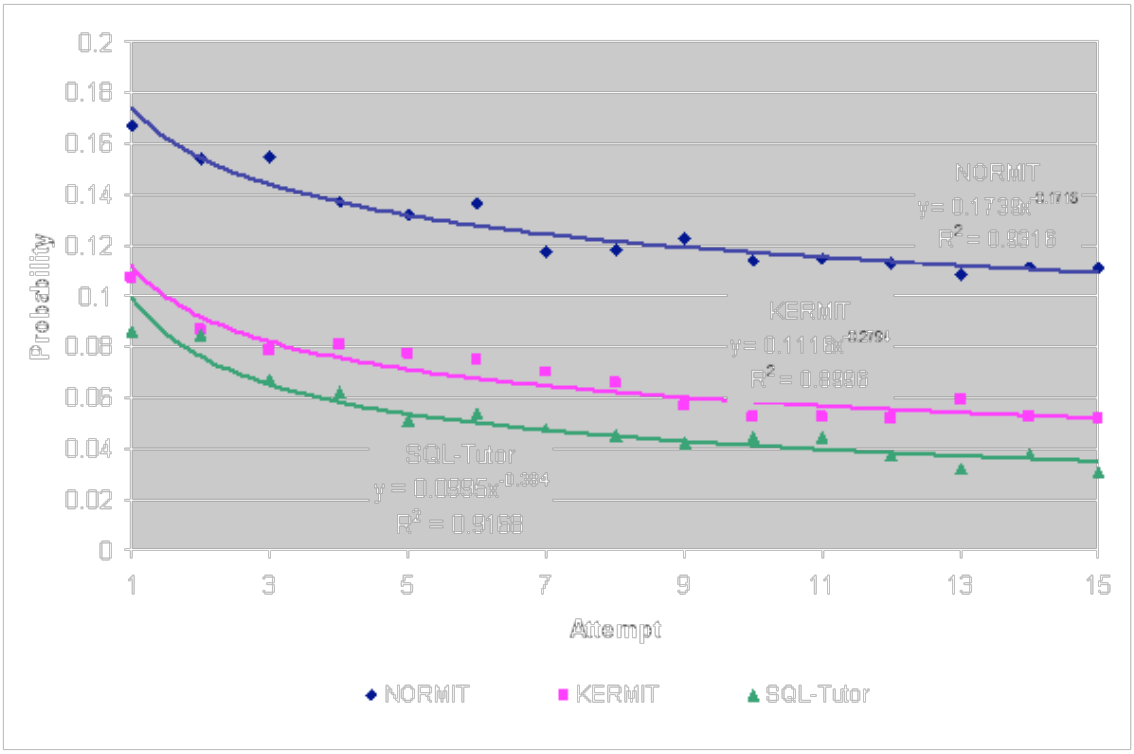
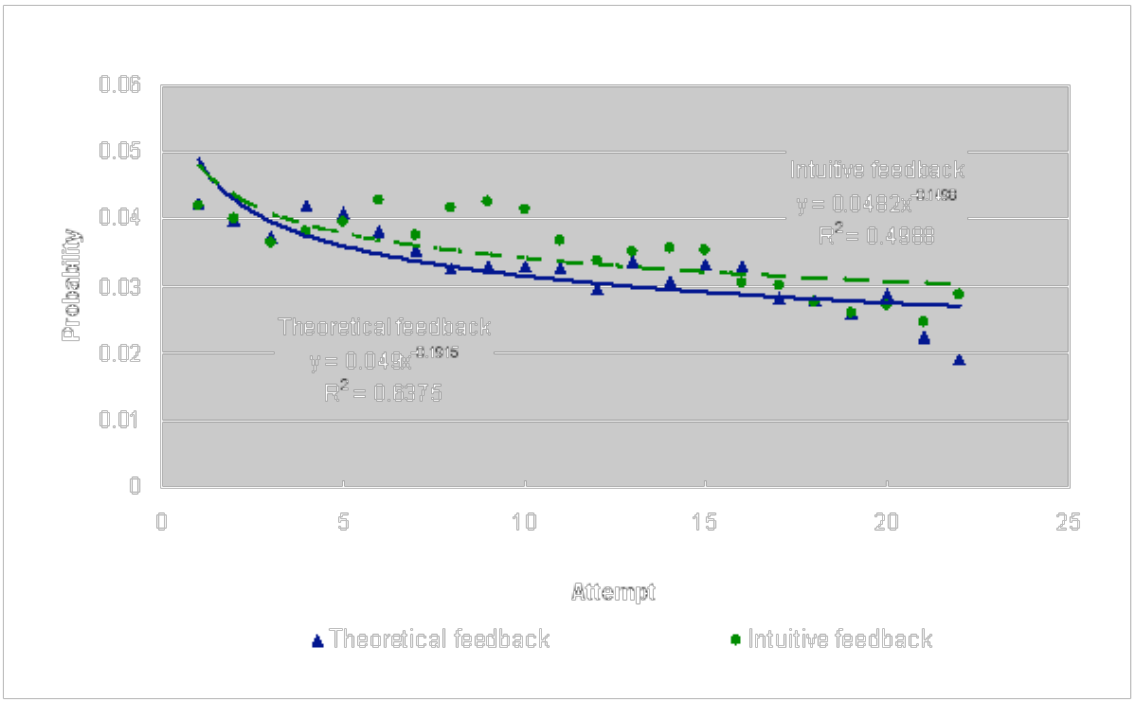

0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Probability
1
3
5
7
9
11
13
15
Attempt
NORMIT
KERMIT
SQL-Tutor

Figure 3


0.06
0.05
0.04
0.03
0.02
0.01
0
Probability
0
5
10
15
20
25
Attempt
Intuitive feedback
y = 0.0482x
R² = 0.4868
Theoretical feedback
y = 0.048x
R² = 0.6375
Theoretical feedback
Intuitive feedback